Installation Instructions

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Copyright (C) 1994, 1995, 1996, 1999, 2000, 2001, 2002, 2004 Free

Software Foundation, Inc.

This file is free documentation; the Free Software Foundation gives

unlimited permission to copy, distribute and modify it.

Prerequisites:

--------------

SOFTSUSY should compile in its ordinary mode with the only prerequisites

being a fortran77 and a C++ compiler.

However, to include the three-loop running with leading two-loop thresholds

for the strong coupling constant and the third generation of fermions within

the MSSM, one also requires:

- pkg-config

- GiNaC >= 1.3.5 (with CLN>=1.3.1)

On Debian/Ubuntu systems one can issue the following command to install

the prerequsites:

sudo apt-get install pkg-config libginac-dev

Quick Installation

------------------

Download the gzipped tarball, from the SOFTSUSY homepage

http://projects.hepforge.org/softsusy/ for instance softsusy-3.5.0.tar.gz. Then,

type (on a linux box)

> tar -xvzf softsusy-3.5.0.tar.gz

> cd softsusy-3.5.0

> ./configure

> make

> make install

which should compile the code and run some sample commands.

If you have trouble with compilation because your machine cannot make the

documentation (because, for example, it doesn't have LaTeX installed), instead

of make you can just do

> make programs

which will not try to install the manuals.

In order to compile with the three-loop running with leading two-loop

thresholds for the strong coupling constant and the third generation of

fermions, replace the ./configure command above with

> ./configure --enable-full\_three\_loop --enable-full\_susy\_threshold

The "make install" command is crucial for the evaluation of two-loop threshold

effects, since corresponding two-loop expressions are stored in the form of

GiNaC archive which could be found by main programs (e.g. softpoint.x) only if

"make install" is issued. After installation, binary files can be found in

PREFIX/bin where PREFIX is the installation prefix specified during

configuration (PREFIX defaults to `/usr/local').

Additional configure switches:

--enable-debug Debug printing (not related to original SOFTSUSY

PRINTOUT option) for the two-loop thresholds and three-loop MSSM RGEs

--prefix Where the installation is done. Notice that the binaries go

into prefix/bin

Since evaluation of two-loop decoupling corrections is rather a time-consuming

process (especially thresholds for b-quark mass) they are not NOT turned on by

default. Three-loop RGEs are also switched off.

Basic Installation

==================

These are generic installation instructions.

The `configure' shell script attempts to guess correct values for

various system-dependent variables used during compilation. It uses

those values to create a `Makefile' in each directory of the package.

It may also create one or more `.h' files containing system-dependent

definitions. Finally, it creates a shell script `config.status' that

you can run in the future to recreate the current configuration, and a

file `config.log' containing compiler output (useful mainly for

debugging `configure').

It can also use an optional file (typically called `config.cache'

and enabled with `--cache-file=config.cache' or simply `-C') that saves

the results of its tests to speed up reconfiguring. (Caching is

disabled by default to prevent problems with accidental use of stale

cache files.)

If you need to do unusual things to compile the package, please try

to figure out how `configure' could check whether to do them, and mail

diffs or instructions to the address given in the `README' so they can

be considered for the next release. If you are using the cache, and at

some point `config.cache' contains results you don't want to keep, you

may remove or edit it.

The file `configure.ac' (or `configure.in') is used to create

`configure' by a program called `autoconf'. You only need

`configure.ac' if you want to change it or regenerate `configure' using

a newer version of `autoconf'.

The simplest way to compile this package is:

1. `cd' to the directory containing the package's source code and type

`./configure' to configure the package for your system. If you're

using `csh' on an old version of System V, you might need to type

`sh ./configure' instead to prevent `csh' from trying to execute

`configure' itself.

Running `configure' takes awhile. While running, it prints some

messages telling which features it is checking for.

2. Type `make' to compile the package.

3. Optionally, type `make check' to run any self-tests that come with

the package.

4. Type `make install' to install the programs and any data files and

documentation.

5. You can remove the program binaries and object files from the

source code directory by typing `make clean'. To also remove the

files that `configure' created (so you can compile the package for

a different kind of computer), type `make distclean'. There is

also a `make maintainer-clean' target, but that is intended mainly

for the package's developers. If you use it, you may have to get

all sorts of other programs in order to regenerate files that came

with the distribution.

Compilers and Options

=====================

Some systems require unusual options for compilation or linking that the

`configure' script does not know about. Run `./configure --help' for

details on some of the pertinent environment variables.

You can give `configure' initial values for configuration parameters

by setting variables in the command line or in the environment. Here

is an example:

./configure CC=c89 CFLAGS=-O2 LIBS=-lposix

\*Note Defining Variables::, for more details.

Compiling For Multiple Architectures

====================================

You can compile the package for more than one kind of computer at the

same time, by placing the object files for each architecture in their

own directory. To do this, you must use a version of `make' that

supports the `VPATH' variable, such as GNU `make'. `cd' to the

directory where you want the object files and executables to go and run

the `configure' script. `configure' automatically checks for the

source code in the directory that `configure' is in and in `..'.

If you have to use a `make' that does not support the `VPATH'

variable, you have to compile the package for one architecture at a

time in the source code directory. After you have installed the

package for one architecture, use `make distclean' before reconfiguring

for another architecture.

Installation Names

==================

By default, `make install' will install the package's files in

`/usr/local/bin', `/usr/local/man', etc. You can specify an

installation prefix other than `/usr/local' by giving `configure' the

option `--prefix=PREFIX'.

You can specify separate installation prefixes for

architecture-specific files and architecture-independent files. If you

give `configure' the option `--exec-prefix=PREFIX', the package will

use PREFIX as the prefix for installing programs and libraries.

Documentation and other data files will still use the regular prefix.

In addition, if you use an unusual directory layout you can give

options like `--bindir=DIR' to specify different values for particular

kinds of files. Run `configure --help' for a list of the directories

you can set and what kinds of files go in them.

If the package supports it, you can cause programs to be installed

with an extra prefix or suffix on their names by giving `configure' the

option `--program-prefix=PREFIX' or `--program-suffix=SUFFIX'.

Optional Features

=================

Some packages pay attention to `--enable-FEATURE' options to

`configure', where FEATURE indicates an optional part of the package.

They may also pay attention to `--with-PACKAGE' options, where PACKAGE

is something like `gnu-as' or `x' (for the X Window System). The

`README' should mention any `--enable-' and `--with-' options that the

package recognizes.

For packages that use the X Window System, `configure' can usually

find the X include and library files automatically, but if it doesn't,

you can use the `configure' options `--x-includes=DIR' and

`--x-libraries=DIR' to specify their locations.

Specifying the System Type

==========================

There may be some features `configure' cannot figure out automatically,

but needs to determine by the type of machine the package will run on.

Usually, assuming the package is built to be run on the \_same\_

architectures, `configure' can figure that out, but if it prints a

message saying it cannot guess the machine type, give it the

`--build=TYPE' option. TYPE can either be a short name for the system

type, such as `sun4', or a canonical name which has the form:

CPU-COMPANY-SYSTEM

where SYSTEM can have one of these forms:

OS KERNEL-OS

See the file `config.sub' for the possible values of each field. If

`config.sub' isn't included in this package, then this package doesn't

need to know the machine type.

If you are \_building\_ compiler tools for cross-compiling, you should

use the `--target=TYPE' option to select the type of system they will

produce code for.

If you want to \_use\_ a cross compiler, that generates code for a

platform different from the build platform, you should specify the

"host" platform (i.e., that on which the generated programs will

eventually be run) with `--host=TYPE'.

Sharing Defaults

================

If you want to set default values for `configure' scripts to share, you

can create a site shell script called `config.site' that gives default

values for variables like `CC', `cache\_file', and `prefix'.

`configure' looks for `PREFIX/share/config.site' if it exists, then

`PREFIX/etc/config.site' if it exists. Or, you can set the

`CONFIG\_SITE' environment variable to the location of the site script.

A warning: not all `configure' scripts look for a site script.

Defining Variables

==================

Variables not defined in a site shell script can be set in the

environment passed to `configure'. However, some packages may run

configure again during the build, and the customized values of these

variables may be lost. In order to avoid this problem, you should set

them in the `configure' command line, using `VAR=value'. For example:

./configure CC=/usr/local2/bin/gcc

will cause the specified gcc to be used as the C compiler (unless it is

overridden in the site shell script).

`configure' Invocation

======================

`configure' recognizes the following options to control how it operates.

`--help'

`-h'

Print a summary of the options to `configure', and exit.

`--version'

`-V'

Print the version of Autoconf used to generate the `configure'

script, and exit.

`--cache-file=FILE'

Enable the cache: use and save the results of the tests in FILE,

traditionally `config.cache'. FILE defaults to `/dev/null' to

disable caching.

`--config-cache'

`-C'

Alias for `--cache-file=config.cache'.

`--quiet'

`--silent'

`-q'

Do not print messages saying which checks are being made. To

suppress all normal output, redirect it to `/dev/null' (any error

messages will still be shown).

`--srcdir=DIR'

Look for the package's source code in directory DIR. Usually

`configure' can determine that directory automatically.

`configure' also accepts some other, not widely useful, options. Run

`configure --help' for more details.